

CSCE 240: Advanced Programming Techniques

Lecture 22: Code Optimization

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

31ST MARCH 2022

Carolinian Creed: “I will practice personal and academic integrity.”

Credits: Some material reused with permission of Dr. Jeremy Lewis. Others used as cited with thanks.

Organization of Lecture 22

- Introduction Section
 - Recap of Lecture 22
 - News / announcements / clarifications
 - TA and SI Updates
- Main Section
 - Memory optimization
 - Runtime optimization
 - Code maintenance ease
 - Task: Project – PA #5 ongoing – check for issues
- Concluding Section
 - About next lecture – Lecture 23
 - Ask me anything

Introduction Section

Recap of Lecture 21

- We looked at memory management
 - Different types of memories available to a program
 - Summary of vector
 - Care to be taken with deletes
- PA5 due April 5, 2022 (Tuesday)

On HW-6

- Considerations
 - Some students are taking extra time
 - Time between PA#5 and PA#6 is tight (Assembling final solution, project report and presentation)
 - Project presentations are crucial for everyone to prepare as well as attend
 - HW-6 will give programming opportunity; help improve best-of-4 score as originally launched
- Polls Participation
 - 8 responded, wanted as optional, for practice
- Decision: We will have HW-6.
 - If people respond, we will grade and consider best of 4 from it (6). Otherwise, best of 4 from 5.
 - Peer review will depend on submissions

Updates from TA, SU

- TA update: Yuxiang Sun (Cherry)
- SI update: Blake Seekings

Main Section

Goals of Programming

- Function goals – meets customer's stated functionality needs
 - Meets user's requirements
 - Meets developer's specifications
- Non-functional goals – has desirable characteristics
 - Runs fast
 - Takes less memory
 - Does not abnormally terminate

Code and Developer Objectives

- **Function goals** – meets customer's stated functionality needs
 - Meets user's requirements
 - Meets developer's specifications
- **Non-functional goals** – has desirable characteristics
 - Runs fast
 - Takes less memory
 - Does not abnormally terminate
 - Is well documented
 - ...

Example: Sorting numbers

- **Input:** a set of N numbers in any order
- **Output:** a set of N numbers, with $a[i-1] \leq a[i]$
- **Function goals** –
 - Gives correct sorted output
 - Handles all given range of inputs
- **Non-functional goals** – has desirable characteristics
 - Runs fast
 - Takes less memory
 - Linear in size of input
 - Does not abnormally terminate
 - Prints output in formatted manner

Code and Developer Objectives

- Writing any program that meets the functional requirements v/s a good code (i.e., scores high in non-functional requirements)
- But meeting all non-functional goals can be hard
 - Space v/s time trade-off
 - In sorting example:
 - Minimize space:
 - Space: N units for N numbers
 - Time: $1 + 2 + \dots + (N)$ operations = $(N * (N+1)) / 2$ in time operations
 - Minimize time:
 - Space: $2N$ units
 - Time: $N \log N$
- Furthermore, you want code to be understandable by others
- Printing of output ...

Example: Sorting numbers

• **Input:** a set of N numbers in any order

• **Output:** a set of N numbers, with $a[i-1] \leq a[i]$

Concept: Memory Optimization

Why Optimize for Memory?

- Unnecessary drag on performance (slow loading, running of program)
- Program may not run on some platforms
 - Mobile phones, games, embedded devices, setup boxes
- Wastage of (natural) resources – storage media, electricity, ...

Reducing Memory Usage

- Use appropriate data type based on range of values possible. Example: int, float, double
- For a group of variables,
 - if size is known,
 - Use data structures (e.g., arrays) of right size
 - Otherwise,
 - dynamic data structures (list) // reduces wastage
(**alternative**: use arrays with a large size; wastes space)
- Do not have unused variables
- Free space when no longer needed

Example: Sorting numbers

•**Input:** a set of N numbers in any order

•**Output:** a set of N numbers, with $a[i-1] \leq a[i]$

For sorting numbers, use array

For sorting strings, use [?]

Concept: Runtime Optimization

Why Optimize for Time?

- Users expect it !
 - One of the motivations for automation/ programming is speed
- Efficient use of computing resources

Reducing Time – Design of Algorithms

Algo 1:

- `current_array = a = Input`
- `While (true)`
 - Check if `current_array` is sorted (i.e., $a[i-1] \leq a[i]$, for $i=1$ to $(N-1)$).
 - If yes,
 - **Return `current_array`**
 - `current_array = Permute (current_array)` (i.e., swap values of any i, j , i not equal j , for $i, j = 1$ to $(N-1)$)

Example: Sorting numbers

- **Input:** a set of N numbers in any order
- **Output:** a set of N numbers, with $a[i-1] \leq a[i]$

Algo 2:

- `current_array = a = Input`
- `For (i=0; i<N; i++) {`
 - `For (j=0; j<N; j++) {`
 - `If(a[i] > a[j])`
 - `Swap(a[i], a[j])`
 - `}`
 - `}`
- **Return `current_array`**

Which one will be efficient ?

Reducing Time – Design of Algorithms

Algo 2:

- `current_array = a = Input`
- `For (i=0; i<N; i++) {`
 - `For (j=0; j<N; j++) {`
 - `If(a[i] > a[j])`
 - `Swap(a[i], a[j])`
 - `}`
- `}`
- `Return current_array`

Algo 3:

- `current_array = a = Input`
- `For (i=0; i<N-1; i++) {`
 - `For (j=1; j<N; j++) {`
 - `If(a[i] > a[j])`
 - `Swap(a[i], a[j])`
 - `}`
- `}`
- `Return current_array`

Example: Sorting numbers

• **Input:** a set of N numbers in any order

• **Output:** a set of N numbers, with $a[i-1] \leq a[i]$

Which one will be efficient ?

Optimizing for Time – Beyond Algorithms

- Input/ Output takes time
 - Use buffering for reading/ writing large data
 - Do not use print in production code
- Choice of data structure to minimize I/O and processing operations
- Advanced methods
 - Look to do processing in parallel
 - Caching of results // storing (full or partial) results for previous invocations

Concept: Code Management Ease

Software Maintenance Considerations

- Others should be able to understand code and change it
 - Documentation of code
 - Meaningful error messages/ prints/ logging
 - Modularity of code
 - Code reuse / usage of functions

Software Maintenance Considerations

- Others should be able to understand code and change it

- Documentation of code
- Meaningful error messages/ prints/ logging
- Modularity of code
- Code reuse / usage of functions

-> Increases development time

-> Can slow execution time

-> Can increase memory at runtime,
development time

-> Increases / decreases development
time

Class Exercise – 10 Mins

- **Objective:** Sorting student records

Type	Memory Consideration	Runtime Considerations	Maintenance Considerations
Number (SSNs)			
Strings (Names – F, M, L)			
Grades			
Overall: LastName + Grade			

Discussion: Course Project

Course Project – Assembling of Prog. Assignments

- **Project:** Develop collaborative assistants (chatbots) that offer innovative and ethical solutions to real-world problems ! *(Based on competition - <https://sites.google.com/view/casy-2-0-track1/contest>)*
- Specifically, **the project will be building a chatbot that can answer questions about a South Carolina member of state legislature from:**
<https://www.scstatehouse.gov/member.php?chamber=H>
 - Each student will choose a district (from 122 available).
 - Programming assignment programs will: (1) extract data from the district, (2) process it, (3) make content available in a command-line interface, (4) handle any user query and (5) report on interaction statistics.

Core Programs Needed for Project

- Prog 1: extract data from the district [\[prog1-extractor\]](#)
- Prog 2: process it (extracted data) based on questions [\[prog2processor\]](#)
- Prog 3: make content available in a command-line interface [\[prog3-ui\]](#)
- Prog 4: handle any user query [\[prog4-userintent2querymapper\]](#)
- Prog 5: report statistics on interaction of a session, across sessions [\[prog5-sessionlogger\]](#)

Objective in Programming Assignment # 5:

Record what happens in a chat session and provide summary

- A user may interact with your chatbot for one question or twenty. How did your chatbot do?
- **Record chat your system makes with each user and report on user session as well total usage statistics (since the chatbot was created)**

Approach Suggested

- Under data folder,
 - have a sub-folder called **chat_sessions**
 - When a person starts a chat session (i.e., starts your program and until does not quit), create a file with the “<data>_<time>.txt” as the name. Save the user’s utterance and the system’s reply there in the order they come. Close this file when the user session ends.
 - Calculate statistics: # user_utterance, #system_utterance and time duration of session
 - have a file called **chat_statistics.csv**.
 - Have a header with columns: S.No, chat_file, # user_utterance, #system_utterance and time taken
 - For each chat file in chat_sessions, there will be a row with the chat statistics you have calculated

Objective in Programming Assignment # 5:

Record what happens in a chat session and provide summary

Approach Suggested

- Under data folder,
 - have a sub-folder called **chat_sessions**
 - When a person starts a chat session (i.e., starts your program and until does not quit), create a file with the “<data>_<time>.txt” as the name. Save the user’s utterance and the system’s reply there in the order they come. Close this file when the user session ends.
 - Calculate statistics: # user_utterance, #system_utterance and time duration of session
 - have a file called **chat_statistics.csv**.
 - Have a header with columns: S.No, chat_file, # user_utterance, #system_utterance and time taken
 - For each chat file in chat_sessions, there will be a row with the chat statistics you have calculated

- Goal: report statistics on interaction of a session, across sessions [Name: **prog5-sessionlogger**]
- One can invoke it with arguments
 - **prog5-sessionlogger –summary**
 - There are 12 chats to date with user asking 23 times and system respond 24 times. Total duration is 456 seconds.
 - **prog5-sessionlogger –showchat-summary 2**
 - Chat 2 has user asking 2 times and system respond 2 times. Total duration is 4 seconds.
 - **prog5-sessionlogger –showchat 2**
 - Chat 2 chat is:
...
...
 - **prog5-sessionlogger –showchat 200**
 - ERROR: there are only 12 chat sessions. Please choose a valid number.

Programming Assignment # 5

- Code organization
 - Create a folder in your GitHub called “**prog5-sessionlogger**”
 - Have sub-folders: src (or code), data, doc, test
 - Have data directory as shown in previous slide
 - ./data/chat_sessions/
 - ./data/chat_statistics.csv
 - Write a 1-page report in ./doc sub-folder
 - Put a log of system interacting in ./test
 - Send a confirmation that code is done by updating Google sheet; optionally, send email to instructor and TA
- Use concepts learned in class
 - Exceptions
 - File operations
 - Dynamic memory

Class Exercise – 10 Mins

- **Objective:** Course Project

Type	Memory Consideration	Runtime Considerations	Maintenance Considerations
Prog 1: [prog1-extractor]			
Prog 2: [prog2processor]			
Prog 3: [prog3-ui]			
[prog4-userintent2querymapper]			
Prog 5: [prog5-sessionlogger]			

Concluding Section

Lecture 22: Concluding Comments

- We discussed code optimization considerations
 - Memory optimization
 - Runtime optimization
 - Code maintenance ease
- Looked at examples
 - Sorting
 - Project

About Next Lecture – Lecture 23

Lecture 23: Advanced: Templates

- Templates
- Class Templates
- Function Templates

20	Mar 24 (Th)	Advanced: Operator overloading	Prog 4 - end
21	Mar 29 (Tu)	Advanced: Memory Management	Prog 5 - start
22	Mar 31 (Th)	Advanced: Code efficiency	
23	Apr 5 (Tu)	Advanced: Templates	Prog 5 – end, Optional HW-6
24	Apr 7 (Th)	AI / ML and Programming	Prog 6 - assembling
25	Apr 12 (Tu)	Review material for Quiz 2	Project due
26	Apr 14 (Th)	In class test	Quiz 2 – In class